# Malware Analysis for Machine Learning

Generated by Doxygen 1.8.11

# Contents

# Chapter 1

# Malware Analysis for Machine Learning

## Getting Started

These instructions will get you a copy of the project up and running on your local machine for development and testing purposes.

### Prerequisites

Requires boost libraries, boost filesystem module and a compiler supporting c++11 standards

```
1 sudo apt-get install libboost-all-dev
```

### Installing

Create the build directory if not present at ROOT

```
1 cd ROOT/
2 mkdir build
```

then

```
1 cd build
2 cmake ..
3 make
4 cd ..
```

You can also run the following commands directly into the ROOT folder, but it'll flood it with unwanted files

```
1 cmake ./
2 make
```

## Running the programs

The two following classifiers are generate in the ROOT directory
I assume that the training set are located here too, otherwise, the default paths can be changed with options

#### malware detection

To use the **malware detector** with default arguments, run

```
1 cd ROOT/
2 ./detector
```

For help and detailled options, run

```
1 ./detector help
```

#### malware classification

To use the **malware familiy classifier** with default arguments, run

```
1 cd ROOT/
2 ./classifier
```

For help and detailled options, run

```
1 ./classifier help
```

## Author

**Aurelien BEC** - *A Machine Learning homework* - AurelBec

## License

This project is licensed under free licence

# Chapter 2

# Hierarchical Index

## 2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

# Chapter 3

# Class Index

## 3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

# Chapter 4

# File Index

## 4.1 File List

Here is a list of all files with brief descriptions:

# Chapter 5

# Class Documentation

## 5.1 Class Class Reference

Parent Classifier.

`#include <VectorSpace.hpp>`

### Public Member Functions

- Class (const std::string &name)
  *Constructor.*
- Class (const std::string &name, const std::set< int > &sets)
  *Constructor.*
- ~Class ()
  *Destructor.*
- void study (const boost::filesystem::path &filePath)
  *Study a manifest.*
- const std::string & name () const
  *Get the name of the Class.*
- const uint nbManifests () const
  *Get the size of the Class.*
- void setNbValue ()
  *Set the number of value in the Class.*
- void setProbability (const uint total)
  *Set the probability of occurence of the Class.*
- const long double getLogProbability (const std::string &prefix, const std::string &value) const
  *Get the natural logarithmic probability of occurence of a value.*
- const long double getLogProbability () const
  *Get the probability of occurence of the Class.*

### Private Member Functions

- const int getSetId (const std::string &prefix) const
  *Get the id of a SubSet.*

### Private Attributes

- uint m_nbMan

  *Number of manifests studied during training phase.*
- uint m_nbValue

  *Number of value studied during training phase.*
- long double m_pL

  *Natural logarithmic probability of Class' occurence.*
- std::vector< SubSet * > m_sets

  *Sets considered by the classifer.*
- const std::string m_name

  *Name of the Class.*

### Friends

- std::ostream & operator<< (std::ostream &out, const Class &c)

## 5.1.1 Detailed Description

Parent Classifier.

This class is used to create a classifier

## 5.1.2 Constructor & Destructor Documentation

### 5.1.2.1 Class::Class ( const std::string & *name* ) [inline]

Constructor.

Constructor of the class Class

**Parameters**

| | |
|---|---|
| *name* | The name of the Class |

### 5.1.2.2 Class::Class ( const std::string & *name*, const std::set< int > & *sets* ) [inline]

Constructor.

Only constructor of the class SubSet

**Parameters**

| | |
|---|---|
| *name* | The name of the Class |
| *sets* | The ids of the SubSet the Class have to considered |

#### 5.1.2.3 Class::~Class ( ) [inline]

Destructor.

Destructor of the class Class

### 5.1.3 Member Function Documentation

#### 5.1.3.1 const long double Class::getLogProbability ( const std::string & *prefix*, const std::string & *value* ) const

Get the natural logarithmic probability of occurence of a value.

Gets the probability of occurence of a value in the Class

**Parameters**

| | |
|---|---|
| *prefix* | The set in which look for the value |
| *value* | The value to look for |

**Returns**

The natural logarithmic probability of occurence of a value

#### 5.1.3.2 const long double Class::getLogProbability ( ) const [inline]

Get the probability of occurence of the Class.

Returns the probability of occurence of the Class

**Returns**

The probability of occurence of the Class

#### 5.1.3.3 const int Class::getSetId ( const std::string & *prefix* ) const [private]

Get the id of a SubSet.

Returns the id of a SubSet

**Parameters**

| | |
|---|---|
| *prefix* | The name of the SubSet to look for |

**Returns**

The id of the SubSet

#### 5.1.3.4 const std::string& Class::name ( ) const [inline]

Get the name of the Class.

Returns the name of the Class

**Returns**

The name of the Class

#### 5.1.3.5 const uint Class::nbManifests ( ) const [inline]

Get the size of the Class.

Returns the size of the Class, the number of manifest studied

**Returns**

The size of the Class

#### 5.1.3.6 void Class::setNbValue ( ) [inline]

Set the number of value in the Class.

Sets the number of value studied in the Class

#### 5.1.3.7 void Class::setProbability ( const uint *total* ) [inline]

Set the probability of occurence of the Class.

Sets the probability of the occurence of the Class

#### 5.1.3.8 void Class::study ( const boost::filesystem::path & *filePath* )

Study a manifest.

Learns data from a manifest during the training phase

**Parameters**

| | |
|---|---|
| *filePath* | The path to the manifest |

CLASS method's definitions

### 5.1.4 Friends And Related Function Documentation

#### 5.1.4.1 std::ostream& operator<< ( std::ostream & *out,* const Class & *c* ) [friend]

### 5.1.5 Member Data Documentation

#### 5.1.5.1 const std::string Class::m_name [private]

Name of the Class.

#### 5.1.5.2 uint Class::m_nbMan [private]

Number of manifests studied during training phase.

#### 5.1.5.3 uint Class::m_nbValue [private]

Number of value studied during training phase.

#### 5.1.5.4 long double Class::m_pL [private]

Natural logarithmic probability of Class' occurence.

#### 5.1.5.5 std::vector<SubSet*> Class::m_sets [private]

Sets considered by the classifer.

The documentation for this class was generated from the following files:

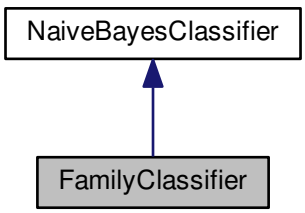
- VectorSpace.hpp
- VectorSpace.cpp

## 5.2 FamilyClassifier Class Reference

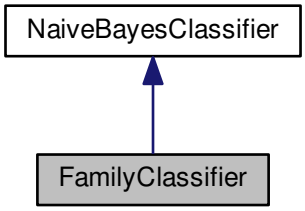
Malware family classifier (inherited from NaiveBayesClassifier )

`#include <Classifiers.hpp>`

Inheritance diagram for FamilyClassifier:



Collaboration diagram for FamilyClassifier:




### Public Member Functions

- FamilyClassifier (const boost::filesystem::path &pDS, const boost::filesystem::path &pHD, const std::set< int > &sets, const uint percent)

  *Constructor.*
- ~FamilyClassifier ()

  *Destructor.*

### Private Member Functions

- void train ()

  *Train.*
- void evaluate (const uint nb, const uint to, const uint from) const

  *Evaluate.*

### Additional Inherited Members

#### 5.2.1 Detailed Description

Malware family classifier (inherited from NaiveBayesClassifier )

This class is used to create a malware family classifier

#### 5.2.2 Constructor & Destructor Documentation

##### 5.2.2.1 FamilyClassifier::FamilyClassifier ( const boost::filesystem::path & *pDS,* const boost::filesystem::path & *pHD,* const std::set< int > & *sets,* const uint *percent* )

Constructor.

Constructor of the inherited class FamilyClassifier

**Parameters**

| | |
|---|---|
| *pDS* | RELATIVE path to the directory containing all the examples |
| *pHD* | RELATIVE path to the SHA1 HASH file, containing name and family of known malware |
| *sets* | The ids of the sets considered by the classifier |
| *percent* | Percent of all availbale examples used for training |

FAMILIY CLASSIFIER method's definitions

#### 5.2.2.2 FamilyClassifier::~FamilyClassifier ( ) [inline]

Destructor.

Destructor of the inherited class FamilyClassifier

### 5.2.3 Member Function Documentation

#### 5.2.3.1 void FamilyClassifier::evaluate ( const uint *nb*, const uint *to*, const uint *from* ) const [private], [virtual]

Evaluate.

Classifies nb examples in the domain selected and show results

**Parameters**

| | |
|---|---|
| *nb* | Number of examples to classify |
| *to* | Position of the first file of the domain |
| *from* | Position of the last file of the domain |

(0 <= nb <= from - to)

Implements NaiveBayesClassifier.

#### 5.2.3.2 void FamilyClassifier::train ( ) [private], [virtual]

Train.

Train our classifier with m_sizeTrain examples

Implements NaiveBayesClassifier.

The documentation for this class was generated from the following files:

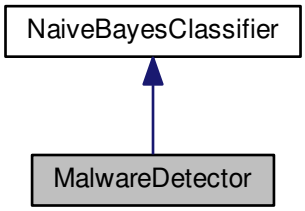
- Classifiers.hpp
- Classifiers.cpp

## 5.3 MalwareDetector Class Reference

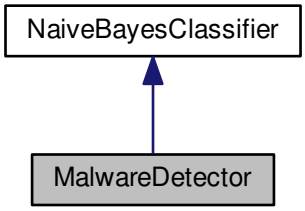
Malware detector (inherited from NaiveBayesClassifier )

`#include <Classifiers.hpp>`

Inheritance diagram for MalwareDetector:




Collaboration diagram for MalwareDetector:




### Public Member Functions

- MalwareDetector (const boost::filesystem::path &pDS, const boost::filesystem::path &pHD, const std::set< int > &sets, const uint percent)

  *Constructor.*
- ~MalwareDetector ()

  *Destructor.*

### Private Member Functions

- void train ()

  *Train.*
- void evaluate (const uint nb, const uint to, const uint from) const

  *Evaluate.*

## Additional Inherited Members

### 5.3.1 Detailed Description

Malware detector (inherited from NaiveBayesClassifier )

This class is used to create a malware detector

### 5.3.2 Constructor & Destructor Documentation

#### 5.3.2.1 MalwareDetector::MalwareDetector ( const boost::filesystem::path & *pDS,* const boost::filesystem::path & *pHD,* const std::set< int > & *sets,* const uint *percent* )

Constructor.

Constructor of the inherited class MalwareClassifier

**Parameters**

| | |
|---|---|
| *pDS* | RELATIVE path to the directory containing all the examples |
| *pHD* | RELATIVE path to the SHA1 HASH file, containing name and family of known malware |
| *sets* | The ids of the SubSets considered by the classifier |
| *percent* | Percent of all availbale examples used for training |

MALWARE DETECTOR method's definitions

#### 5.3.2.2 MalwareDetector::~MalwareDetector ( ) [inline]

Destructor.

Destructor of the inherited class MalwareClassifier

### 5.3.3 Member Function Documentation

#### 5.3.3.1 void MalwareDetector::evaluate ( const uint *nb,* const uint *to,* const uint *from* ) const [private], [virtual]

Evaluate.

Classifies nb examples in the domain selected and show results

**Parameters**

| | |
|---|---|
| *nb* | Number of examples to classify |
| *to* | Position of the first file of the domain |
| *from* | Position of the last file of the domain |

(0 <= nb <= from - to)

Implements NaiveBayesClassifier.

#### 5.3.3.2 void MalwareDetector::train ( ) [private], [virtual]

Train.

Train our classifier with m_sizeTrain examples

Implements NaiveBayesClassifier.

The documentation for this class was generated from the following files:

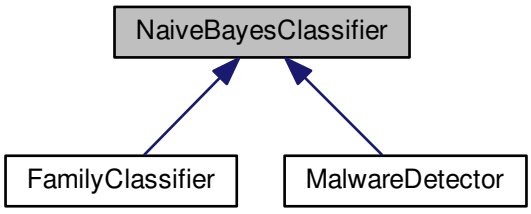
- Classifiers.hpp
- Classifiers.cpp

## 5.4 NaiveBayesClassifier Class Reference

Parent Classifier.

`#include <Classifiers.hpp>`

Inheritance diagram for NaiveBayesClassifier:




### Public Member Functions

- NaiveBayesClassifier (const boost::filesystem::path &pDS, const boost::filesystem::path &pHD, const std::set< int > &sets, const uint percent)

  *Constructor.*
- virtual ~NaiveBayesClassifier ()

  *Destructor.*
- void evaluate (const uint nbTests) const

  *Classify.*

### Protected Member Functions

- virtual void train ()=0

  *Train.*
- virtual void evaluate (const uint nb, const uint to, const uint from) const =0

  *Evaluate.*
- virtual const std::string getClass (const boost::filesystem::path &filePath) const

  *getClass*

### Protected Attributes

- const uint m_percent

  *Percent of all availbale examples used for training.*
- uint m_total

  *Total of all the examples available.*
- uint m_sizeTrain

  *Maximum number of examples to train our classifier (= total ∗ percent)*
- uint m_sizeTest

  *Maximum number of examples to test our classifier (= total - train)*
- const boost::filesystem::path m_pathHashDico

  *RELATIVE path to the SHA1 HASH file, containing name and family of known malware.*
- const boost::filesystem::path m_pathDataSet

  *RELATIVE path to the directory containing all the examples.*
- std::set< std::string > m_hashMalware

  *All the name of known malware (read from m_pathHashDico)*
- std::vector< Class ∗ > m_class

  *Contains each class the classifier study.*
- const std::set< int > m_sets

  *Contains the id of the SubSets considered by the classifier.*

## 5.4.1 Detailed Description

Parent Classifier.

This class is used to create a classifier

## 5.4.2 Constructor & Destructor Documentation

### 5.4.2.1 NaiveBayesClassifier::NaiveBayesClassifier ( const boost::filesystem::path & *pDS*, const boost::filesystem::path & *pHD*, const std::set< int > & *sets*, const uint *percent* )

Constructor.

Constructor of the parent class NaiveBayesClassifier

**Parameters**

| | |
|---|---|
| *pDS* | RELATIVE path to the directory containing all the examples |
| *pHD* | RELATIVE path to the SHA1 HASH file, containing name and family of known malware |
| *sets* | The ids of the sets considered by the classifier |
| *percent* | Percent of all availbale examples used for training |

NAIVE BAYES CLASSIFIER method's definitions

#### 5.4.2.2 virtual NaiveBayesClassifier::~NaiveBayesClassifier ( ) [inline], [virtual]

Destructor.

Vitrual Destructor of the parent class NaiveBayesClassifier

### 5.4.3 Member Function Documentation

#### 5.4.3.1 void NaiveBayesClassifier::evaluate ( const uint *nbTests* ) const

Classify.

Classifies examples in the know and/or unkwon domain

**Parameters**

| | |
|---|---|
| *nbTests* | number of examples to classify |

#### 5.4.3.2 virtual void NaiveBayesClassifier::evaluate ( const uint *nb,* const uint *to,* const uint *from* ) const [protected], [pure virtual]

Evaluate.

Classifies nb examples in the domain selected and show results

**Parameters**

| | |
|---|---|
| *nb* | Number of examples to classify |
| *to* | Position of the first file of the domain |
| *from* | Position of the last file of the domain |

(0 <= nb <= from - to)

Implemented in MalwareDetector, and FamilyClassifier.

#### 5.4.3.3 const string NaiveBayesClassifier::getClass ( const boost::filesystem::path & *filePath* ) const [protected], [virtual]

getClass

Returns the most likely class of the example

**Parameters**

| | |
|---|---|
| *filePath* | RELATIVE path to the example |

**Returns**

The most likely class

#### 5.4.3.4 virtual void NaiveBayesClassifier::train ( ) [protected], [pure virtual]

Train.

Train our classifier with m_sizeTrain examples

Implemented in MalwareDetector, and FamilyClassifier.

### 5.4.4 Member Data Documentation

#### 5.4.4.1 std::vector<Class*> NaiveBayesClassifier::m_class [protected]

Contains each class the classifier study.

#### 5.4.4.2 std::set<std::string> NaiveBayesClassifier::m_hashMalware [protected]

All the name of known malware (read from m_pathHashDico)

#### 5.4.4.3 const boost::filesystem::path NaiveBayesClassifier::m_pathDataSet [protected]

RELATIVE path to the directory containing all the examples.

#### 5.4.4.4 const boost::filesystem::path NaiveBayesClassifier::m_pathHashDico [protected]

RELATIVE path to the SHA1 HASH file, containing name and family of known malware.

#### 5.4.4.5 const uint NaiveBayesClassifier::m_percent [protected]

Percent of all availbale examples used for training.

#### 5.4.4.6 const std::set<int> NaiveBayesClassifier::m_sets [protected]

Contains the id of the SubSets considered by the classifier.

#### 5.4.4.7 uint NaiveBayesClassifier::m_sizeTest [protected]

Maximum number of examples to test our classifier (= total - train)

#### 5.4.4.8 uint NaiveBayesClassifier::m_sizeTrain [protected]

Maximum number of examples to train our classifier (= total * percent)

#### 5.4.4.9 uint NaiveBayesClassifier::m_total [protected]

Total of all the examples available.

The documentation for this class was generated from the following files:

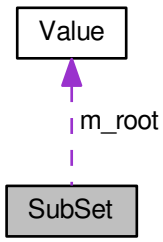
- Classifiers.hpp
- Classifiers.cpp

## 5.5 SubSet Class Reference

class SubSet

`#include <VectorSpace.hpp>`

Collaboration diagram for SubSet:




### Public Member Functions

- SubSet (const int id)
  *Constructor.*
- ~SubSet ()
  *Destructor.*
- void add (const std::string &value)
  *Add a value to the graph.*
- const int getId () const
  *Get id.*
- const std::string & name () const
  *Get name.*
- const uint getOcc (const std::string &value)
  *Get occurence.*
- const uint getOcc () const
  *Get id.*
- const long double getLogProbability (const std::string &value)
  *Get natural logarithmic probability.*

### Static Public Member Functions

- static const std::string & name (const uint id)

  *Get name.*

### Private Attributes

- const int m_id

  *Id of the SubSet.*
- uint m_nbValue

  *Number of values in the SubSet.*
- long double m_nbLValue

  *Natural logarithmic number of value.*
- Value m_root

  *First node of the graph of Values.*

### Static Private Attributes

- static const std::vector< std::string > setNames

  *Stored the name of an id corresponding to a SubSet.*

### Friends

- std::ostream & operator<< (std::ostream &out, const SubSet &ss)

  *Operator <<.*

## 5.5.1 Detailed Description

class SubSet

This class represents a set Contains the id of the set and a graph with all the value read in the examples which belong to this set

## 5.5.2 Constructor & Destructor Documentation

#### 5.5.2.1 SubSet::SubSet ( const int *id* ) [inline]

Constructor.

Only constructor of the class SubSet

#### 5.5.2.2 SubSet::~SubSet ( ) [inline]

Destructor.

Destructor of the class SubSet

## 5.5.3 Member Function Documentation

#### 5.5.3.1 void SubSet::add ( const std::string & *value* )

Add a value to the graph.

Adds a new value to the graph which start at m_root

**Parameters**

| | |
|---|---|
| *value* | The value to add |

SUBSET method's definitions

#### 5.5.3.2 const int SubSet::getId ( ) const [inline]

Get id.

Returns the id of the SubSet

**Returns**

The id of the SubSet

#### 5.5.3.3 const long double SubSet::getLogProbability ( const std::string & *value* ) [inline]

Get natural logarithmic probability.

Returns the natural logarithmic propobability of occurence of a value in the SubSet

**Parameters**

| | |
|---|---|
| *value* | The value to check the probability |

**Returns**

The natural logarithmic propobability of occurence of a value

#### 5.5.3.4 const uint SubSet::getOcc ( const std::string & *value* )

Get occurence.

Returns the occurence of specified value in the SubSet

**Parameters**

| | |
|---|---|
| *value* | The value to return the occurence |

**Returns**

The occrence of the value

#### 5.5.3.5 const uint SubSet::getOcc ( ) const [inline]

Get id.

Returns the sum of occurence of all the values in the SubSet

**Returns**

The sum of occurence

#### 5.5.3.6 const std::string& SubSet::name ( ) const [inline]

Get name.

Returns the name of the SubSet

**Returns**

The name of the SubSet

#### 5.5.3.7 static const std::string& SubSet::name ( const uint *id* ) [inline], [static]

Get name.

Returns the name of a SubSet

**Parameters**

| | |
|---|---|
| *id* | The set's id to return the name |

**Returns**

The name of a SubSet

### 5.5.4 Friends And Related Function Documentation

#### 5.5.4.1 std::ostream& operator<< ( std::ostream & *out,* const SubSet & *ss* ) [friend]

Operator <<.

Overloads the << operator to print a Value

### 5.5.5 Member Data Documentation

#### 5.5.5.1 const int SubSet::m_id [private]

Id of the SubSet.

#### 5.5.5.2 long double SubSet::m_nbLValue [private]

Natural logarithmic number of value.

#### 5.5.5.3 uint SubSet::m_nbValue [private]

Number of values in the SubSet.

#### 5.5.5.4 Value SubSet::m_root [private]

First node of the graph of Values.

#### 5.5.5.5 const vector< string > SubSet::setNames [static], [private]

**Initial value:**

```
=
    {"Hardware components",
     "Requested permissions",
     "App components",
     "Filtered intents",
     "Restricted API calls",
     "Used permissions",
     "Suspicious API calls",
     "Network addresses"}
```

Stored the name of an id corresponding to a SubSet.

The documentation for this class was generated from the following files:

- VectorSpace.hpp
- VectorSpace.cpp

## 5.6 Value Class Reference

class Value

`#include <VectorSpace.hpp>`

### Public Member Functions

- Value ()

  *Constructor.*
- Value (const std::string &value)

  *Constructor.*
- ~Value ()

  *Destructor.*
- void addOcc ()

  *Add occurence.*
- const uint getOcc () const

  *Get occurence.*
- void addNext (const std::string &value)

  *Add a value to the graph.*
- Value * goNext (const std::string &value)

  *Go to next.*
- const std::string & getValue () const

  *Get value.*
- const bool isInNext (const std::string &value) const

  *Check if the value exists in the successors.*

### Private Member Functions

- void print (std::ostream &out, const std::string offset) const

  *Print the value.*

### Private Attributes

- uint m_occ

  *Number of times the value appear in the SubSet.*
- const std::string m_value

  *String stored in the Value.*
- std::vector< Value * > m_next

  *Adresses of all the following Values.*

### Friends

- std::ostream & operator<< (std::ostream &out, const Value &v)

  *Operator <<.*

#### 5.6.1 Detailed Description

class Value

This class is used to represents the node of a graph

### 5.6.2 Constructor & Destructor Documentation

#### 5.6.2.1 Value::Value ( ) [inline]

Constructor.

Default constructor of the class Value

#### 5.6.2.2 Value::Value ( const std::string & *value* ) [inline]

Constructor.

Constructor of the class Value

**Parameters**

| | |
|---|---|
| *value* | value contained by the node |

#### 5.6.2.3 Value::~Value ( ) [inline]

Destructor.

Destructor of the class Value Delete recursively all the Value of the SubSet

### 5.6.3 Member Function Documentation

#### 5.6.3.1 void Value::addNext ( const std::string & *value* ) [inline]

Add a value to the graph.

Adds a new value to the successors of the current node

**Parameters**

| | |
|---|---|
| *value* | The value to add |

#### 5.6.3.2 void Value::addOcc ( ) [inline]

Add occurence.

Adds one to the number of value's occurence

#### 5.6.3.3 const uint Value::getOcc ( ) const [inline]

Get occurence.

Returns the number of times the value appear in the SubSet

**Returns**

true if added, false if not

#### 5.6.3.4 const std::string& Value::getValue ( ) const [inline]

Get value.

Returns the value of the current node

**Returns**

value

#### 5.6.3.5 Value * Value::goNext ( const std::string & *value* ) [inline]

Go to next.

Returns the adress of the next Value containing value

**Parameters**

| | |
|---|---|
| *value* | The value to look for in the nexts |

**Returns**

Adress of the next Value, NULL if it doesn't exist

#### 5.6.3.6 const bool Value::isInNext ( const std::string & *value* ) const [inline]

Check if the value exists in the successors.

Checks if a value is in the successors of the current node

**Parameters**

| | |
|---|---|
| *value* | The value to check |

**Returns**

true if yes, false if not

VALUE method's definitions

#### 5.6.3.7 void Value::print ( std::ostream & *out,* const std::string *offset* ) const [private]

Print the value.

Prints recursively the value and all the nexts

### 5.6.4 Friends And Related Function Documentation

#### 5.6.4.1 std::ostream& operator<< ( std::ostream & *out,* const Value & *v* ) [friend]

Operator <<.

Overloads the << operator to print a Value

### 5.6.5 Member Data Documentation

#### 5.6.5.1 std::vector<Value*> Value::m_next [private]

Adresses of all the following Values.

#### 5.6.5.2 uint Value::m_occ [private]

Number of times the value appear in the SubSet.

#### 5.6.5.3 const std::string Value::m_value [private]

String stored in the Value.

The documentation for this class was generated from the following files:

- VectorSpace.hpp
- VectorSpace.cpp

# Chapter 6

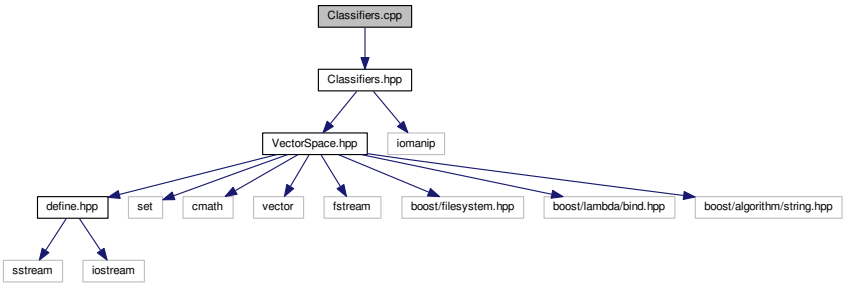
# File Documentation

## 6.1 Classifiers.cpp File Reference

Contains the implemented methods of the different classifiers.

```
#include "Classifiers.hpp"
```

Include dependency graph for Classifiers.cpp:




### 6.1.1 Detailed Description

Contains the implemented methods of the different classifiers.

**Author**

Aurelien Bec

**Date**

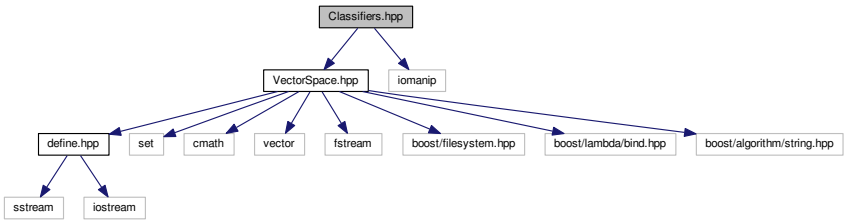
November 17th 2017

## 6.2 Classifiers.hpp File Reference

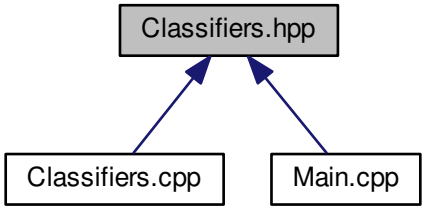
Contains the definition of different classifiers.

```
#include "VectorSpace.hpp"
#include <iomanip>
```

Include dependency graph for Classifiers.hpp:




This graph shows which files directly or indirectly include this file:




### Classes

- class NaiveBayesClassifier

  *Parent Classifier.*
- class FamilyClassifier

  *Malware family classifier (inherited from NaiveBayesClassifier )*
- class MalwareDetector

  *Malware detector (inherited from NaiveBayesClassifier )*

### 6.2.1 Detailed Description

Contains the definition of different classifiers.

**Author**

Aurelien Bec

**Date**

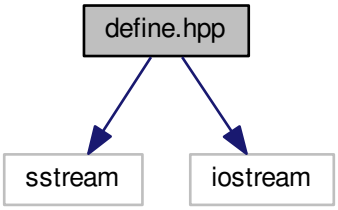
November 17th 2017

## 6.3 define.hpp File Reference

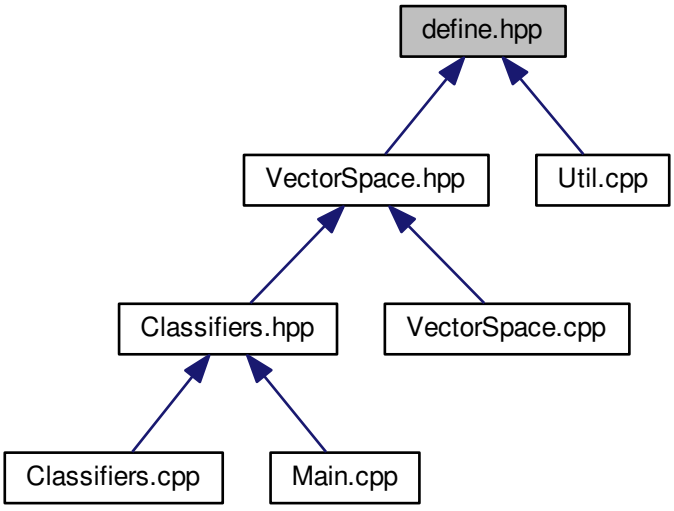
Contains the global definitions for the project.

```
#include <sstream>
#include <iostream>
```

Include dependency graph for define.hpp:




This graph shows which files directly or indirectly include this file:




### Macros

- #define DEBUG_ON false
- #define BAD_BOUND 30
- #define GOOD_BOUND 20
- #define DEFAULT_D_PERCENT 66

- #define DEFAULT_D_NBTESTS 1000
- #define DEFAULT_C_PERCENT 66
- #define DEFAULT_C_NBTESTS 2000
- #define DEFAULT_MINNUMBERSAMPLE 20
- #define DEFAULT_SETS {1, 2, 3, 4, 5, 6, 7, 8}
- #define DEFAULT_PATH_HASHDICO "drebin/sha256_family.csv"
- #define DEFAULT_PATH_DATASET "drebin/feature_vectors"
- #define TOKENIZERS "\t :;,.!()?->/%=_"

### Functions

- const std::string stringNULL ("0")
- const std::string colorizedValue (const double value, const bool invert)
- const std::string toLower (const std::string &in)
- const long double divide (const long double a, const long double b)

### 6.3.1 Detailed Description

Contains the global definitions for the project.

**Author**

Aurelien Bec

**Date**

November 17th 2017

### 6.3.2 Macro Definition Documentation

#### 6.3.2.1 #define BAD_BOUND 30

#### 6.3.2.2 #define DEBUG_ON false

#### 6.3.2.3 #define DEFAULT_C_NBTESTS 2000

#### 6.3.2.4 #define DEFAULT_C_PERCENT 66

#### 6.3.2.5 #define DEFAULT_D_NBTESTS 1000

#### 6.3.2.6 #define DEFAULT_D_PERCENT 66

#### 6.3.2.7 #define DEFAULT_MINNUMBERSAMPLE 20

#### 6.3.2.8 #define DEFAULT_PATH_DATASET "drebin/feature_vectors"

#### 6.3.2.9 #define DEFAULT_PATH_HASHDICO "drebin/sha256_family.csv"

#### 6.3.2.10 #define DEFAULT_SETS {1, 2, 3, 4, 5, 6, 7, 8}

#### 6.3.2.11 #define GOOD_BOUND 20

#### 6.3.2.12 #define TOKENIZERS "\t :;,.!()?->/%=_"

### 6.3.3 Function Documentation

#### 6.3.3.1 const std::string colorizedValue ( const double *value,* const bool *invert* )

#### 6.3.3.2 const long double divide ( const long double *a,* const long double *b* )

#### 6.3.3.3 const std::string stringNULL ( "0" )

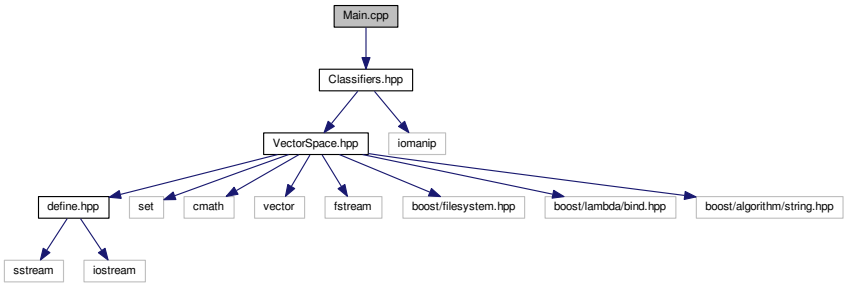
#### 6.3.3.4 const std::string toLower ( const std::string & *in* )

## 6.4 Main.cpp File Reference

Contains the main function.

```
#include "Classifiers.hpp"
```

Include dependency graph for Main.cpp:




### Functions

- int main (int argc, char ∗argv[ ])

### 6.4.1 Detailed Description

Contains the main function.

**Author**

Aurelien Bec

**Date**

November 17th 2017

### 6.4.2 Function Documentation

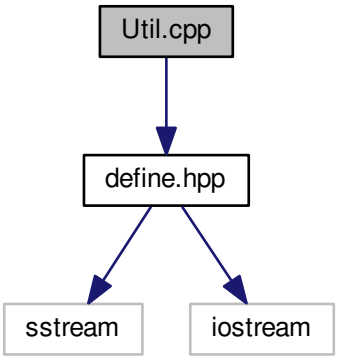
#### 6.4.2.1 int main ( int *argc*, char * *argv[]* )

## 6.5 README.md File Reference

## 6.6 Util.cpp File Reference

Contains some useful functions for the project.

`#include "define.hpp"`
Include dependency graph for Util.cpp:




### Functions

- const string toLower (const string &in)
- const string colorizedValue (const double value, const bool invert)
- const long double divide (const long double a, const long double b)

### 6.6.1 Detailed Description

Contains some useful functions for the project.

**Author**

Aurelien Bec

**Date**

November 17th 2017

### 6.6.2 Function Documentation

#### 6.6.2.1 const string colorizedValue ( const double *value,* const bool *invert* )

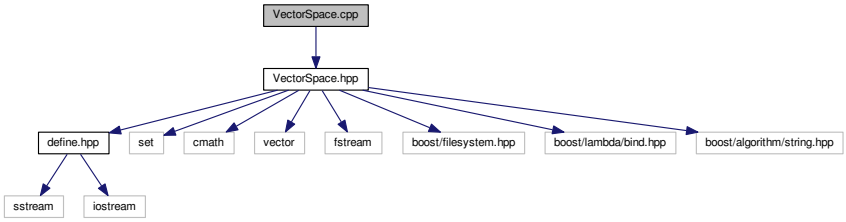
#### 6.6.2.2 const long double divide ( const long double *a,* const long double *b* )

#### 6.6.2.3 const string toLower ( const string & *in* )

## 6.7 VectorSpace.cpp File Reference

Contains the implemented methods of VectorSpace and the other class need.

`#include "VectorSpace.hpp"`
Include dependency graph for VectorSpace.cpp:




### Functions

- ostream & operator<< (ostream &out, const Class &c)
- ostream & operator<< (ostream &out, const SubSet &ss)
- ostream & operator<< (ostream &out, const Value &v)

### 6.7.1 Detailed Description

Contains the implemented methods of VectorSpace and the other class need.

**Author**

Aurelien Bec

**Date**

November 17th 2017

### 6.7.2 Function Documentation

#### 6.7.2.1 ostream& operator<< ( ostream & *out,* const Class & *c* )

printing method's definitions

#### 6.7.2.2 ostream& operator<< ( ostream & *out,* const SubSet & *ss* )

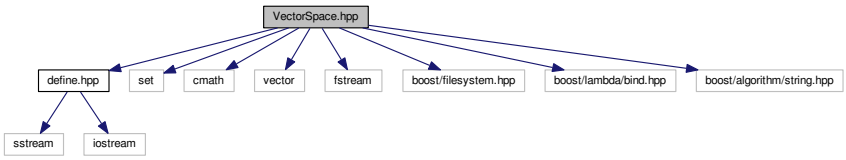
#### 6.7.2.3 ostream& operator<< ( ostream & *out,* const Value & *v* )

## 6.8 VectorSpace.hpp File Reference

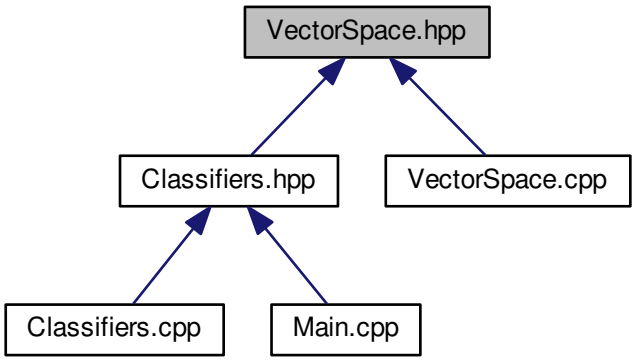
Contains the definition of VectorSpace and the other class need.

```
#include "define.hpp"
#include <set>
#include <cmath>
#include <vector>
#include <fstream>
#include <boost/filesystem.hpp>
#include <boost/lambda/bind.hpp>
#include <boost/algorithm/string.hpp>
```

Include dependency graph for VectorSpace.hpp:




This graph shows which files directly or indirectly include this file:




### Classes

- class Value

  *class Value*
- class SubSet

  *class SubSet*
- class Class

  *Parent Classifier.*

### 6.8.1 Detailed Description

Contains the definition of VectorSpace and the other class need.

**Author**

Aurelien Bec

**Date**

November 17th 2017

# Index